Moving mesh finite difference schemes for the porous medium equation

G. Scherer and M. J. Baines

Abstract

This is a report of work in progress on numerical experiments using finite difference schemes designed to solve nonlinear diffusion equations on a dynamic mesh. A semi-implicit and an implicit finite difference scheme are defined for the porous medium equation in 1D. The moving mesh is constructed by imposing a conservative distribution principle. The resulting equations are solved in terms of the grid coordinates, and the underlying PDE solution is computed algebraically *a posteriori* from the conserved mass on this mesh.

Key words: Dynamic meshes, moving boundaries, porous medium

1 Introduction

We consider solutions of the porous medium equation in 1D with *compact* support and *symmetric* initial values.

1.1 Porous medium equation in 1D

Definition 1 *The porous medium equation (PME) in 1D is the non-linear diffusion equation:*

$$u_t = (u^m u_x)_x, \quad \textit{where } m > 0. \tag{1}$$

It arises for example in models of gas diffusion through porous media under Darcy's law which relates velocity to pressure gradients.

We choose initial conditions at time $t = t_0$ to be $u(x, t_0) \equiv u_0(x)$, positive on the interval $(x_-(t_0), x_+(t_0))$, and $u(x, t_0) = 0$ on the boundaries. It can be proved that under these conditions a solution $u(x,t)$ exists for all $t \geq t_0$ with free boundaries at $x_-(t)$ and $x_+(t)$. The boundaries propagate with a finite speed given by

$$v(x_\pm) = -\frac{1}{m}\{u(x_\pm,t)^m\}_x$$

The solution of the porous medium equation has some interesting features, in particular the integral of u (mass) and the centre of mass are constant in time:

Lemma 2 *Invariance of mass and centre of mass in time.*

The integral of u (or mass) is conserved in time. If $I = \int_{x_-}^{x_+} u(x,t)dx$ is the total mass, then

$$\frac{dI}{dt} = \int_{x_-(t)}^{x_+(t)} u_t dx = \int_{x_-}^{x_+} (u^m u_x)_x dx = u^m u|_{x_-(t)}^{x_+(t)} = 0. \tag{2}$$

Also, if $\overline{x}$ is the centre of mass scaled by the (constant) total mass,

$$\frac{d\overline{x}}{dt} = \frac{d}{dt}\int_{x_-}^{x_+} xudx = \int_{x_-(t)}^{x_+(t)} x(u^m u_x)_x dx$$

$$= [xu^m u_x]_{x_-(t)}^{x_+(t)} - \int_{x_-(t)}^{x_+(t)} (u^m u_x)dx = -\frac{1}{m+1}u^m|_{x_-(t)}^{x_+(t)} = 0.$$

- Note that for symmetric initial values, the solution remains symmetric, so that $u_x(0,t) = 0$, $\forall t$.
- The value of m influences the speed and the character of the solution close to the moving boundary. If $m = 1$ the local solution is quadratic, while if $m > 1$ it is a steep front with a consequently slower displacement.
- The *waiting time,* i.e. the delay before the boundary points start moving outwards, depends on $1/m$, the cut being $1/m = 1$, for which there is no delay.

1.2 Self-similar solutions

When concentrating on the qualitative aspects of partial differential equations (PDE) the concepts of scaling invariance and the associated self-similar solutions become important (see e.g. [5]).

Definition 3 ***Scaling invariance property**: Given a system (u,x,t) satisfying a PDE and a mapping to a new system $(\widehat{u},\widehat{x},\widehat{t})$ under the power-law transformation*

$$u = \lambda^\gamma \widehat{u}\ ,\ x = \lambda^\beta\ \widehat{x},\ t = \lambda^\alpha \widehat{t} \tag{3}$$

where λ is an arbitrary positive scale parameter, the original (u, x, t) system is said to be **scaling invariant** *if the* PDE *under consideration is identical in both the original and the transformed co-ordinates. For simplicity, α is often chosen to be $\alpha = 1$.*

In other words, the form of the equation does not change if the variables are scaled.

Self-similar solutions, i.e. solutions of the PDE which are themselves invariant under the same scaling are obtained by choosing $\lambda = t$. They reveal an important property of the phenomenon described by the PDE: the phenomenon reproduces itself on different time/space scales, or in other words, the spatial distributions of the characteristics of the phenomenon $u(x, t)$, although varying in time, remain geometrically similar. Formalizing this:

Definition 4 *([3]) A solution $u = u(x, t)$ of a non-linear evolution partial differential equation is called* ***self-similar*** *if the knowledge of u at the instant of time t_0 is sufficient to obtain u for all $t > t_0$ by a suitable re-scaling, i.e., there exist time-dependent scales t^γ and t^β such that, measured in these scales, the phenomenon becomes time independent:*

$$u(x, t) = t^\gamma \; f\left(\frac{x}{t^\beta}\right). \tag{4}$$

An important by-product of self-similarity is that an independent variable, t, is "lost" and instead of a time-dependent PDE one only has to solve an ordinary differential equation (ODE).

Introducing the transformations (3) into the differential equation (1) and the mass conservation integral (2), one obtains that the scaling invariance conditions for the PME, the parameters β, γ being

$$\beta = \frac{1}{2+m}, \; \gamma = -\frac{1}{2+m}. \tag{5}$$

To obtain a self-similar solution for the PME we substitute the expression

$$\frac{u}{t^\gamma} = f\left(\frac{x}{t^\beta}\right)$$

into the PME using the known values of β and γ. To simplify the notation introduce $w \equiv u/t^\gamma$, $y \equiv x/t^\beta$. After the necessary differentiations, one obtains the differential equation (ODE)

$$\gamma t^{\gamma-1} f - \beta t^{\gamma-1} y f' = t^{\gamma-1}(f^m f')'$$

which reduces to

$$-\beta y f = f^m f'$$

using the boundary condition $u(0) = f(0) = 0$. The solution is:

$$f(y) = c(1-y^2)^{\frac{1}{m}}, \text{ with } c = \left(\frac{m}{2(2+m)}\right)^{\frac{1}{m}}.$$

Hence, the self-similar solution of the PME is:

$$u(x,t) = ct^{\gamma}\left[1-\left(\frac{x}{t^{\beta}}\right)^2\right]^{\frac{1}{m}} \tag{6}$$

with $|x|/t^{\beta} < 1$ defining the compact support.

1.3 Mesh movement for the PME

We shall use an r-refinement approach, i.e. maintain the number of nodes globally but relocate them according to the behaviour of the solution. For that purpose the PDE is coupled to a mesh evolution control which defines a velocity and the resulting equations, when solved, define simultaneously the mesh and the physical solution.

We will assume symmetric initial values, hence consider now the PME only on the right half-interval of its compact support, the centre point (and from now on the left-hand boundary) remaining fixed at $x = 0$, i.e., $\dot{x} = 0$, $\forall t$.

The adaptive mesh is defined by means of an invertible transformation between the physical and the computational coordinates at a given time t. Let x and ξ denote the physical and computational coordinates respectively, with $x \in [0, x_+(t)]$, and $\xi \in [0, N]$, N a positive integer. A one-to-one coordinate transformation between these domains is denoted at time t by

$$x = \widehat{x}(\xi, t),\ \xi \in [0, N] \quad \text{with} \quad \widehat{x}(0,t) = 0,\ \widehat{x}(x_+(t), t) = N. \tag{7}$$

The Jacobian of the mapping is $\widehat{x}_{\xi}$.

To control the mesh evolution the basic idea is a *conservative distribution principle (CDP)* using a monitor function $M(u)$ depending on the solution u of the PDE. If, as we do in this report, we choose $M(u) = u$, the CDP implies that the area under the solution u in any subinterval $(\widehat{x}_1(t), \widehat{x}_2(t))$ of the compact support $(\widehat{x}_-(t), \widehat{x}_+(t))$ will be independent of time, as in

$$\int_{\widehat{x}_1(t)}^{\widehat{x}_2(t)} u(x', t)\,dx' = c(\widehat{x}_1, \widehat{x}_2) \tag{8}$$

This conservation principle allows us to recast the PME as a *mesh velocity equation*. The Eulerian form of the conservation law (8) is

$$u_t + (uv)_x = 0$$

from which it follows that the PME (1) can be restated as the purely spatial equation

$$-(uv)_x = (u^m u_x)_x$$

which after integration gives

$$uv = -u^m u_x$$

(recalling that $u_x(0,t) = 0$ when $v(0,t) = 0$). Hence, provided that $u \neq 0$,

$$v = -u^{m-1} u_x = -\frac{1}{m}(u^m)_x \tag{9}$$

Now, under the inverse of the mapping (7) the left hand side of equation (8) can be expressed as

$$\int_{\xi_1}^{\xi_2} \widehat{u}(\xi', t) x'_\xi d\xi', \tag{10}$$

independent of t, where $\widehat{u}(\xi,t) = u(\widehat{x}(\xi,t),t)$. Since ξ_1, ξ_2 are arbitrary it follows that there exists a function $\widehat{c}(\xi)$ such that

$$\widehat{u}(\xi)\widehat{x}_\xi = u(\widehat{x}(\xi,t),t)\widehat{x}_\xi = \widehat{c}(\xi). \tag{11}$$

From (9) the velocity $\widehat{v}(\xi,t) = v(\widehat{x}(\xi,t),t)$ under the transformation can be written in terms of ξ and $\widehat{c}(\xi)$ only as

$$\widehat{v}(\xi,t) = -\frac{(\widehat{c}(\xi))^{m-1}}{(\widehat{x}_\xi)^m}\frac{\partial}{\partial\xi}\left(\frac{\widehat{c}(\xi)}{\widehat{x}_\xi}\right) = -\frac{1}{m}\frac{\widehat{c}(\xi)}{\widehat{x}_\xi}\frac{\partial}{\partial\xi}\left(\frac{\widehat{c}(\xi)}{\widehat{x}_\xi}\right)^m. \tag{12}$$

or with explicit derivatives:

$$\widehat{v}(\xi,t) = \left(\frac{\widehat{c}(\xi)}{\widehat{x}_\xi}\right)^{m-1}\left(\frac{\widehat{c}(\xi)}{\widehat{x}_\xi^3}\widehat{x}_{\xi\xi} - \frac{1}{\widehat{x}_\xi^2}\frac{\partial\widehat{c}(\xi)}{\partial\xi}\right). \tag{13}$$

1.4 Discretization

It is convenient to drop the hat notation at this point. Suppose that a uniform mesh, with $\Delta\xi = 1$, is defined on the computational domain by

$$\xi_i = i, \ \ i = 0, 1, ...N,$$

and denote the corresponding mesh points in the x space by $\{x_0, x_1, ...x_N\}$, where x_0 is fixed but the x_i, ($i = 1, ...N$), vary with t. Introducing local masses $c_{i+1/2}(t)$ for each of the subintervals defined by the mesh points $\{x_0, x_1, ..., x_N\}$, the CDP (8) then implies local mass conservation in the form

$$\int_{x_i(t)}^{x_{i+1}(t)} u(x', t)dx' = c_{i+1/2}, \qquad (i = 0, ..., N-1), \tag{14}$$

independent of t (see [4,2,6]). The constants $c_{i+1/2}$ can be obtained *a priori* from the initial data.

Equation (12) may be discretized as

$$v_i = -\frac{1}{m}\frac{c_i}{\Delta x_i}\left[\left(\frac{c_{i+1/2}}{x_{i+1}-x_i}\right)^m - \left(\frac{c_{i-1/2}}{x_i - x_{i-1}}\right)^m\right]. \tag{15}$$

where Δx_i approximates x_ξ at the point i.

Equation (15) is particularly simple in the case of the so-called local mass *equidistribution principle (EP)*, see [7], which assumes that

$$c_{i+1/2}(t) = \int_{x_i(t)}^{x_{i+1}(t)} udx = \frac{1}{N}\int_{x_0}^{x_N} udx, \; i = 0, ...N-1,$$

i.e. the mass in each cell is uniform in "space" as well as independent of time: $c_{i+1/2}(t) \equiv c$. Since these uniform masses are maintained in time, the velocity (15) at any time t becomes

$$v_i = -\frac{1}{m}\frac{c^{m+1}}{(x_{i+1}-x_{i-1/2})}\left[\left(\frac{1}{x_{i+1}-x_i}\right)^m - \left(\frac{1}{\Delta x_i}\right)^m\right]. \tag{16}$$

These velocities relocate the meshpoints in such a way that, as the material diffuses, the cells retain their original masses $c_{i+1/2}$.

Note that due to the fact that, by construction $\Delta\xi = 1$, conveniently only the nodal values x_i appear in the discretization of these equations.

Having solved the mesh equations for the x_i, the PDE solution u_i can be obtained from the conservation equation (14).

2 Numerical algorithms

2.1 General moving mesh algorithm

The basic steps of a moving mesh algorithm based on the CDP are as follows:

Algorithm 1 *General moving mesh algorithm*

(0) At time t_0, define an initial mesh with N nodes, $X(t_0) = \{x_0, x_1, ..., x_N\}$, where $x_0 = 0$, $x_N = x_+(t_0)$. Compute the solution $u(t_0, x_i)$, $i = 0, ...N$, from the initial conditions, select a numerical integration method and calculate the local masses c_i.

Step foward in time in steps Δt:

(i) Using a finite difference scheme calculate the new mesh values $X(t + \Delta t)$ using a mesh movement equation. Note that x_0 does not change in time.

(ii) Based on the local mass conservation principle, recover the values of $u(t + \Delta t, x)$ from the numerical integration formula chosen at stage (0), except for the interface point x_N where the boundary condition applies: $u(x_N, t) = 0$, $\forall t$.

The advantage that the simpler, moving mesh equations, for EP meshes have over the equations for the more general CDP meshes, is balanced by the need to generate an initial mesh with equidistributed local mass. In addition, given the form of the PME similarity solution, equidistribution produces a higher meshpoint density close to x_0 than for the rest of the interval, in particular there are relatively few nodes close to the *moving* boundary.

At different stages of the algorithm we need numerical methods for the approximation of the integrals and the differential equations and for the solution of nonlinear system of equations. These will now be discussed.

2.2 Mass integral approximations

Numerical approximations of the local mass integrals in each cell $[x_i, x_{i+1}]$, $i = 0, ...N - 1$, are needed twice, first in the initial step of the algorithm to define the cell masses c_i, and subsequently, at each time step, to recover the values of the PME solution $u(t + \Delta t, x)$ from the CDP.

There are several possible approximation techniques:

(1) The one-interval trapezium rule: $\int_{x_i}^{x_{i+1}} u(t,x)dx \approx \frac{1}{2}(u_{i+1}+u_i)(x_{i+1}-x_i)$.
(2) The trapezium-type rule over two intervals:

$$\int_{x_{i-1}}^{x_{i+1}} u(t,x)dx \approx \frac{1}{2}(u_{i+1}+u_i)(x_{i+1}-x_i) + \frac{1}{2}(u_i+u_{i-1})(x_i-x_{i-1}).$$

(3) The lower order form $\frac{1}{2}\ u_i\ (x_{i+1}-x_i) \approx \int_{x_i}^{x_{i+1}(t)} u(t,x)dx$

As mentioned before, once the mesh movement differential equation has been integrated to give the new nodal positions $x_i(t+\Delta t)$, the $u_i(t+\Delta t)$ on the new mesh are *recovered* using the corresponding local mass integral formula. The trapezium rule (1) and the upwind difference (3) approximations use the fact that the solution u_N is zero at x_N in order that the values of u_i at the other mesh points can be computed explicitly. In the trapezium formula case the $u_i(t+\Delta t)$ are linked together, in fact the vector $u_i(t+\Delta t)$, $i=0...N$ can be written as the solution of a bidiagonal system of equations with condition number $O(N)$: it is therefore subject to possible larger errors than if using the formula (3).

When applying the general *CDP* mesh (not the *EP* one), in addition to the approximation for the local mass $c_i \sim \int_{x_i(t)}^{x_{i+1}(t)} u(t,x)dx$, the derivative $c_i' \equiv \frac{\partial c}{\partial \xi}|_{x_i}$ must be computed. A central difference is used to approximate this derivative:

$$c_i' \sim \frac{c_{i+1}-c_{i-1}}{2} \qquad (i=1,...N-1) \tag{17}$$

At $i=0$, based on the symmetry of the solution u in $[x_-,x_+]$ we use $c_0'=0$.

In order to define the mesh at a new time step, three finite difference schemes involving different mesh movement equations were tested, first a simple explicit scheme, then a semi-implicit, and finally an implicit method.

2.3 The explicit scheme

The basic explicit Euler scheme, implemented for comparison reasons, uses the equation (9) in the form: $v_i = -u_i^{m-1} u_{x|_{x_i}}$. Here, $u_{x|_{x_i}}$ is approximated in the interior by a forward finite difference, while for u_i the mean value at two consecutive nodes is used $\frac{1}{2}(u_i+u_{i+1})$, and for v_i also a mean $\frac{1}{2}(v_{i-1}+v_i)$, so that in computing v_i all of u_i, u_{i-1}, u_{i+1}, are involved. At x_N the u_x is obtained from a simple one-sided backward difference.

For both the semi-implicit and the implicit scheme it is more convenient to use equation (12) or (13) as they involve the nodes directly and not through the physical solution u.

2.4 A semi-implicit scheme

A semi-implicit node-order-preserving scheme satisfying a maximum/minimum principle for non-contracting regions [1] based on equation (12) is defined, starting with a semi-implicit backward Euler scheme,

$$\frac{x_i^{n+1}-x_i^n}{\Delta t} = -\frac{1}{m}\frac{c_i}{(\Delta x)_i^n}\left[\left(\frac{c_{i+1/2}}{x_{i+1}^{n+1}-x_i^{n+1}}\right)^m - \left(\frac{c_{i-1/2}}{x_i^{n+1}-x_{i-1}^{n+1}}\right)^m\right] \tag{18}$$

where $c_{i\pm1/2}$ denotes a lagged cell mass in a cell between two gridpoints x_i^n and $x_{i\pm1}^n$. Here, x_i^n is the coordinate of the i'th gridpoint at the n'th timestep, and $(\Delta x)_i^n$ is the forward finite difference in x at the same timestep.

For the boundary nodes, since $x_0 = 0$ is fixed, no equation is needed. The right boundary values $x_N^{n+1}$are estimated from an explicit one-sided difference scheme applied to the second form of equation (9),

$$\frac{x_N^{n+1}-x_N^n}{\Delta t} = -\frac{1}{m}\left(\frac{-(u_{N-1}^n)^m}{x_N^n - x_{N-1}^n}\right) \tag{19}$$

The scheme (18) can be rewritten as,

$$x_i^{n+1} - x_i^n = -d_{i-1/2}(x_i^{n+1}-x_{i-1}^{n+1}) + d_{i+1/2}(x_{i+1}^{n+1}-x_i^{n+1}) \tag{20}$$

where the $d_{i\pm1/2}$ are the coefficients,

$$d_{i+1/2} = \frac{1}{m}\frac{\Delta t}{(c_{i+1/2}\Delta x)_i^{n+1}}\frac{(c_{i-1/2})^m}{(x_{i+1}^{n+1}-x_i^{n+1})(x_i^{n+1}-x_{i-1}^{n+1})^m},$$

and

$$d_{i-1/2} = \frac{1}{m}\frac{c_{i+1/2}\Delta t}{(\Delta x)_i^{n+1}}\frac{(c_{i+1/2})^m}{(x_{i+1}^{n+1}-x_i^{n+1})^m(x_i^{n+1}-x_{i-1}^{n+1})}.$$

To solve for the x_i^{n+1}, $i = 1, ...N-1$, a functional iteration can be defined based on equation (20). Starting with the initial iterate $\mathbf{x}^{(0)} = (x_1^n, ...x_{N-1}^n)$, a *linear* tridiagonal system is solved:

$$-d_{i-1/2}^{(k)}x_{i-1}^{(k+1)} + (1 + d_{i-1/2}^{(k)} + d_{i+1/2}^{(k)})x_i^{(k+1)} - d_{i+1/2}^{(k)}x_{i+1}^{(k+1)} = x_i^{(k)},$$

where $d_{i\pm1/2}^{(k)}$ are the $d_{i\pm1/2}$ now evaluated at the $\mathbf{x}^{(k)}$ obtained at the last functional iteration. The value of the right boundary node x_N is also updated at each iteration.

[1] The proof is by *reductio ab absurdum*

Note that if the original mesh is ordered, the fact that the mesh is a node-order-preserving mesh insures that no tangling of the mesh occurs, provided of course that the updating of the right boundary node is consistent with the ordering.

In order to smooth the solution values at the interfaces, after the grid points at level $n+1$, (time $t+\Delta t$) have been computed the value of x_{N-1}^{n+1} is recomputed as a weighted mean of its neighbours: $x_{N-1}^{n+1} = \frac{1}{4}x_{N-2}^{n+1} + \frac{1}{4}x_{N-1}^{n+1} + \frac{1}{4}x_N^{n+1}$. This smoothing still preserves the order of the nodes.

Another smoothing was also tested, whereby the values of the solution u_{N-1}^{n+1} are recomputed as a weighted mean, $u_{N-1}^{n+1} = \frac{1}{4}u_{N-2}^{n+1} + \frac{1}{4}u_{N-1}^{n+1} + \frac{1}{4}u_N^{n+1}$.

2.5 An implicit scheme

An *implicit* Euler applied to equation (13) has also been used for $x_1, ..., x_{N-1}$, resulting in the $N-1$ nonlinear equations,

$$\frac{x_i^{n+1} - x_i^n}{\Delta t} = \frac{c_i^m}{(D_0 x_i^{n+1})^{m+2}} D_+ D_- x_i^{n+1} - \frac{c_i' . c_i^{m-1}}{(D_0 x_i^{n+1})^{m+1}}. \tag{21}$$

where D_-, D_0, D_+ are respectively backward, central and forward difference operators.

Again, an equation for the boundary node x_N^{n+1} is needed. Three possibilities were explored, the tests showing that the last one is the most effective choice:

- Defining x_N by an explicit one-sided finite difference for equation (9). This restricts the ratio $\Delta t/\Delta x^2$ for which the mesh remains "untangled".
- Using a one-sided implicit approximation of equation (9) at x_N . This is not a feasible option due to the difficulty in assigning a value to c_N.
- Defining x_N by a one-sided implicit scheme but for the equation (**??**), expressing u as a function of the mass c and x:

$$\frac{x_N^{n+1} - x_N^n}{\Delta t} = -\frac{1}{2m}\left[D_-(u_N^{(n+1)})^m + D_-(u_N^{(n)})^m\right]. \tag{22}$$

One can avoid the use of u completely because, using the PME boundary conditions, $u_N^{(n+1)}$ is set to 0 and $u_{N-1}^{(n+1)}$ is substituted using the expression for the mass approximation, namely:

$$u_{N-1}^{(n+1)} = \frac{2.c_{N-1}}{x_N^{n+1} - x_{N-2}^{n+1}},$$

so that the whole system of equations is expressed only in terms of x's and c's.

The resulting nonlinear systems in $X(t+\Delta t) = \{\ x_i^{n+1}\}\ i = 0...N$ are solved using an iterative algorithm implemented in the user-friendly public domain subroutine HYBRID1 ([8]). The program is based on the Powell hybrid algorithm, a global method for solving a nonlinear system $\mathbf{F}(\mathbf{x}) = \mathbf{0}$, that converges to a solution from almost any starting point. It combines Newton steps on $\mathbf{F}(\mathbf{x})$ with a back-tracking strategy (dogleg) for the associated minimization problem if the new approximate does not reduce the value of $\|\mathbf{F}\|$.

The user must provide a subroutine which calculates the functions only, the Jacobian being approximated by forward differences at the starting point. Two of the main characteristics of HYBRD1 involve the choice of the correction as a convex combination of the Newton and scaled gradient directions, and the updating of the Jacobian by the rank-1 method of Broyden.

The accuracy of HYBRD1 is controlled by the convergence parameter *tol*. Unless high precision solutions are required, the recommended value for *tol* is the square root of the machine precision. The test assumes that the functions are reasonably well behaved. If the stopping condition is satisfied with $tol = 10^{-k}$, then the larger components of the approximately zero $\mathbf{x}$ have k significant decimal digits and INFO is set to 1. There is a danger that the smaller components of $\mathbf{x}$ may have large relative errors, but the fast rate of convergence of HYBRD1 usually avoids this possibility.

Possible difficulties when using the subroutine are:

HYBRID1 calls the more general routine HYBRD. The choice of step length in the forward-difference approximation to the Jacobian assumes that the relative errors in the functions are of the order of the machine precision. If this is not the case, HYBRD1 may fail (usually with INFO = 4). One should then use HYBRD directly, or one of the programs which require the analytic Jacobian (HYBRJ1 and HYBRJ).

In addition, sometimes it is necessary to start from a different point in the domain. Therefore, our starting iterates are chosen from a set of convex combinations of an explicit Euler step for the equation (13) and the mesh at the last time step, $X(t)$. The choice is made based on estimates of the condition number of the jacobians at the possible initial iterates.

3 Numerical examples

3.1 General PME equation of power m

Consider first the general PME equation $u_t = (u^m u_x)_x$ for $m = 3$:

$$u_t = (u^3 u_x)_x, \tag{23}$$

As initial data, the self-similar solution is used at time $t_0 = 0.01$. Its compact support is defined by $|x| \leq 0.01^{0.2} \approx 0.4$. We integrate up to $t = 1$. This problem was also considered in [4,2].

The program **Porousmedium_1** implements the "*implicit*" algorithm for a moving mesh based on a general CDP, i.e., the local masses are conserved in time but are not equal in space. Some details are:

- The initial mesh $\{0 = x_0, x_1, ..., x_+ = x_N\}$ is uniform. After computing the solution u_i at the nodes (with $u_N = 0$, local mass approximations c_i are computed, this time for $i = 1, ...N$. Formula #3 is used for $i = 1, ...N - 1$, and formula #1 for $i = N$. Then the approximations to $c_i(\xi), i = 0, ...N - 1$ are obtained from equation (17).
- For the iteration over time in steps Δt to obtain $u(t+\Delta t)$ HYBRID1 is used to solve the nonlinear system in $x_i(t + \Delta t)$, $i = 1, ...N - 1$. The "special" nodes x_0 and $x_N(t)$ are computed as $x_0 = 0$ with x_N given by the one-sided implicit scheme (22). As mentioned earlier, the initial iterate is defined using a convex combination of an explicit Euler step and the computed grid at the previous step $x(t)$. The condition number of the jacobian at the initial values is computed with the intention eventually to start the iteration at the best conditioned point. For the time being, initial values are tried until the algorithm converges. The tolerance parameter tol is set to $tol = 10^{-6}$, so that the computed solution should have approximately 6 significant digits.

The tables below compare the errors, at a time t_{final} as close as possible to $t = 1$, of the computed solution u, using the semi-implicit scheme with and without smoothing and the implicit method. The results correspond to increasing sizes of Δt and increasing number of nodes, 80 and 120.

In the semi-implicit method the fixed-point iterations stop if the $\|residual\|_2 < 10^{-2}$ or the number of iterations > 20. Changing these values to 10^{-3} and 40 iterations gives essentially the same results.

For N=80

Semi-implicit

Δt	error in u_0	max error in u_i
5.10^{-3}	$0.28.10^{-2}$	$7.03.10^{-2}$
10^{-2}	$0.36.10^{-2}$	$7.75.10^{-2}$
2.10^{-2}	$0.53.10^{-2}$	$9.59.10^{-2}$
5.10^{-2}	$1.15.10^{-2}$	0.25

Semi-implicit with $x-$smoothing

Δt	error in u_0	max error in u_i
5.10^{-3}	$0.23.10^{-2}$	$1.25.10^{-1}$
10^{-2}	$0.35.10^{-2}$	$6.56.10^{-2}$
2.10^{-2}	$0.55.10^{-2}$	$8.6.10^{-2}$
5.10^{-2}	$1.15.10^{-2}$	0.18

Semi-implicit with $u-$smoothing

Δt	error in u_0	max error in u_i
5.10^{-3}	$0.28.10^{-2}$	$1.33.10^{-2}$
10^{-2}	$0.36.10^{-2}$	$2.06.10^{-2}$
2.10^{-2}	$0.54.10^{-2}$	$4.0.10^{-2}$
5.10^{-2}	$1.15.10^{-2}$	0.19

Implicit

Δt	error in u_0	max error in u_i
5.10^{-3}	$0.3293.10^{-3}$	$7.004.10^{-3}$
10^{-2}	$0.1465.10^{-2}$	$3.64.10^{-2}$
2.10^{-2}	$2.097.10^{-2}$	0.2554
5.10^{-2}	overtaking	

For N=120

Semi-implicit

Δt	error in u_0	max error in u_i
10^{-4}	$0.13.10^{-2}$	$5.97.10^{-2}$
10^{-3}	$0.14.10^{-2}$	$6.1.10^{-2}$
10^{-2}	$0.27.10^{-2}$	$7.86.10^{-2}$

Semi-implicit with $x-$smoothing

Δt	error in u_0	max error in u_i
10^{-4}	$0.17.10^{-2}$	$2.23.10^{-1}$
10^{-3}	$4.0.10^{-4}$	$1.75.10^{-1}$
10^{-2}	$0.28.10^{-2}$	$6.86.10^{-2}$

Semi-implicit with u-smoothing

Δt	error in u_0	max error in u_i
10^{-4}	$0.13.10^{-2}$	$1.08.10^{-2}$
10^{-3}	$0.14.10^{-2}$	$1.13.10^{-2}$
10^{-2}	$0.27.10^{-2}$	$2.9.10^{-2}$

Implicit

Δt	error in u_0	max error in u_i
10^{-4}	$0.4838.10^{-3}$	$1.168.10^{-2}$
10^{-3}	$0.2665.10^{-3}$	$1.021.10^{-2}$
10^{-2}	large condition number	

The next table shows the behaviour of the methods for a fixed value of the time step but different number of nodes.

For $\Delta t = 10^{-4}$ at $t_{final} \approx 1$,

Semi-implicit

N	error in u_0	max error in u_i
20	$1.28.10^{-2}$	$7.74.10^{-2}$
40	$0.52.10^{-2}$	$7.21.10^{-2}$
80	$0.21.10^{-2}$	$6.46.10^{-2}$
120	$0.13.10^{-2}$	$5.97.10^{-2}$

Semi-implicit with x-smoothing

N	error in u_0	max error in u_i
20	$2.79.10^{-2}$	$3.75.10^{-1}$
40	$1.04.10^{-2}$	$3.13.10^{-1}$
80	$0.35.10^{-2}$	$2.55.10^{-1}$

Semi-implicit with u-smoothing

N	error in u_0	max error in u_i
20	$1.28.10^{-2}$	$3.21.10^{-2}$
40	$0.52.10^{-2}$	$2.11.10^{-2}$
80	$0.21.10^{-2}$	$1.38.10^{-2}$

Implicit

N	error in u_0	max error in u_i
20	$0.4473.10^{-2}$	$1.554.10^{-2}$
40	$0.1981.10^{-2}$	$1.407.10^{-2}$
80	$0.8261.10^{-3}$	$1.26.10^{-2}$
120	$0.4838.10^{-3}$	$1.168.10^{-2}$

We also run tests for $m = 5$, starting the calculations with the self-similar solution at the same $t = 0.01$ and extending until $t_{final} \approx 1$. The compact support here is defined by $x_{\pm} \approx \pm 0.5$.

For N = 20 with different Δt values:

Semi-implicit

Δt	error in u_0	max error in u_i
10^{-4}	$0.18.10^{-1}$	$9.58.10^{-2}$
10^{-3}	$0.19.10^{-1}$	$9.6.10^{-2}$
10^{-2}	$2.17.10^{-2}$	$1.0.10^{-1}$

Semi-implicit with x-smoothing

Δt	error in u_0	max error in u_i
10^{-4}	$7.37.10^{-2}$	$6.19.10^{-1}$
10^{-3}	$3.48.10^{-2}$	$5.79.10^{-1}$
10^{-2}	$1.07.10^{-2}$	$4.58.10^{-1}$

Semi-implicit with u-smoothing

Δt	error in u_0	max error in u_i
10^{-4}	$0.18.10^{-1}$	$6.96.10^{-2}$
10^{-3}	$0.19.10^{-1}$	$6.96.10^{-2}$
10^{-2}	$2.17.10^{-2}$	$6.96.10^{-2}$

Implicit:

Δt	error in u_0	max error in u_i
10^{-4}	$1.00.10^{-2}$	$2.64.10^{-2}$
10^{-3}	$0.98.10^{-2}$	$2.62.10^{-2}$
10^{-2}	$0.8.10^{-2}$	$2.44.10^{-2}$

Comparing for $\Delta t = 10^{-3}$ with several number of nodes:

Semi-implicit

N	error in u_0	max error in u_i
20	$0.19.10^{-1}$	$9.6.10^{-2}$
40	$0.8.10^{-2}$	$1.0.10^{-1}$
80	$0.38.10^{-2}$	$1.033.10^{-1}$

Semi-implicit with x-smoothing

N	error in u_0	max error in u_i
20	$3.48.10^{-2}$	$5.79.10^{-1}$
40	$0.64.10^{-2}$	$5.0.10^{-1}$
80	$3.44.10^{-4}$	$4.41.10^{-1}$

Semi-implicit with u-smoothing

N	error in u_0	max error in u_i
20	$1.91.10^{-2}$	$6.96.10^{-2}$
40	$0.84.10^{-2}$	$5.52.10^{-2}$
80	$0.38.10^{-2}$	$4.37.10^{-2}$

Implicit

N	error in u_0	max error in u_i
20	$0.9832.10^{-2}$	$2.624.10^{-2}$
40	$0.439.10^{-2}$	$2.723.10^{-2}$
80	$0.1846.10^{-2}$	$2.727.10^{-2}$

For all methods and for $m = 3, 5$, increasing the number of nodes while keeping a common time step Δt results in an, expected, decrease of the errors, or at least very similar errors. For most these runs the computed interface x_N is smaller than the exact at $t_{final} \approx 1$.

The results when keeping instead the number of nodes and varying the time step are more ambiguous. The error for the semi-implicit method with x-smoothing oscilates in several of the tests whereas when using the semi-implicit with $u-$smoothing the errors increase as expected with increasing time step.

The implicit method works very well in some cases, but when the number of nodes is large (120 nodes if one considers the whole compact support), the results are not good.

For comparison, the purely *explicit* finite difference scheme implemented in the program **FD_MBainesMarch09** was used to test the problem with $m = 3$. As in the Porousmedium_1 program, it starts with an uniform mesh and uses Formula #3 for the mass approximations. The preliminary results already indicated that the method was not competitive. The following table shows results for 11 number of nodes.

Running up to a $t_{final} \approx 1$, for $N = 11$,

Δt	error in x_N
10^{-5}	$2.17.10^{-2}$
10^{-4}	$2.17.10^{-2}$
10^{-3}	$2.13.10^{-2}$
2.10^{-3}	$2.13.10^{-2}$
5.10^{-3}	No results

For $N = 21$, the breaking point is already 10^{-4} and for $N = 41$, 10^{-5}.

Accepting that the stability condition is roughly $\Delta t/\Delta x^2 < 1$, there is no sense in continuing with more nodes as the time step has to be too small.

4 Conclusions up to this point

Two types of methods were designed, a semi-implicit and an implicit scheme that both discretize a mesh movement equation expressed in terms of the nodes x and the local masses c. The semi-implicit scheme uses an explicit method for the boundaries, whereas the implicit uses an implicit one-sided method. In order to improve the values computed at the boundaries, two smoothing methods are tried with the semi-implicit scheme: smoothing of the end nodes and smoothing of the solution at the end nodes. It can be proved that for a non-contracting domain the semi-implicit method with $x-$smoothing produces an untangled mesh.

The preliminary testing indicates that although the implicit scheme often produces smaller errors, its behaviour is not so robust as the semi-implicit scheme with u-smoothing.

On the other hand, the expense in using an implicit scheme versus any of the others is much higher. The following table lists the CPU times in seconds of two methods compared for different number of nodes, using a small enough time step so that any of the methods still remains stable. The time relation between the implicit and the semi_implicit scheme (without any smoothing) starts from 150 for $N = 41$ going down to 16 for $N = 11$.

$N, \Delta t$	Semi_implicit	Implicit
11, 5.10^{-4}	0.0156	0.2496
21, 5.10^{-5}	0.156	9.05
41, 5.10^{-5}	0.31	47

But this is not a fair comparison because the accuracy of the results obtained are not the same. The following comparison is more meaningful: for each number of nodes we compare the "best run" for each method, i.e., the one with the smallest error for x_N using the largest time step, in other words the smallest number of computations.

$N = 11$	error in x_N	Δt	CPU time in sec.
Semi-implicit	$2.13.10^{-2}$	5.10^{-3}	$1.56.10^{-2}$
Implicit	$0.6168.10^{-3}$	2.10^{-2}	$3.12.10^{-2}$
$N = 21$			
Semi-implicit	$0.2086.10^{-2}$	5.10^{-4}	$6.24.10^{-2}$
Implicit	$0.1247.10^{-2}$	2.10^{-2}	$6.24.10^{-2}$
$N = 41$			
Semi-implicit	$0.5522.10^{-3}$	1.10^{-4}	0.1716
Implicit	$0.8079.10^{-4}$	1.10^{-2}	0.4524

The work relation varies from 34 times better accuracy for double the CPU time in the $N = 11$ case, to twice the accuracy for the same CPU time for the $N = 21$. For $N = 41$, there is again a good advantage for the implicit scheme.

5 Bibliograpy

References

[1] M. J. Baines, M. E. Hubbard, P. K. Jimack, A.C. Jones, *Scale-invariant moving finite elements for nonlinar partial differential equations in two dimensions*, Applied Numerical Mathematics, **56,** pp. 230-252, 2006.

[2] M. J. Baines, M. E. Hubbard and P. K. Jimack. *Velocity-based moving mesh methods for nonlinear partial differential equations*, Communications in Computational Physics **10**, pp. 509-576, 2011.

[3] G. I. Barenblatt, *Scaling*, CUP, 2003.

[4] K. W. Blake and M. J. Baines, *A moving mesh method for non-linear parabolic problems*, Numerical Analysis Report 2/02, Department of Mathematics, University of Reading, UK, 2002.

[5] C. J. Budd, G. J. Collins, W. Z. Huang and R. D. Russell, *Self-similar numerical solutions of the porous medium equation using moving mesh methods,* Philosophical Trans. : Mathematical, Physical and Engineering Sciences 357, Geometric Integration: Numerical Solution of Differential Equations on Manifolds (1999), 1047-1077.

[6] T. E. Lee, *Modelling Time-dependent Partial differential equations using a moving mesh approach based on conservation,* PhD thesis, University of Reading, UK, (2011).

[7] W. Huang, Y. Ren and R. Russell, *Moving Mesh Partial Differential Equations (MMPDES) based on the equidistribution principle,* SIAM J. Numer. Anal., vol. 31, No. 3, pp. 709-730, June 1994.

[8] www.netlib.org